C. Alcaraz, R. Roman, and J. Lopez, "Análisis de primitivas criptográficas para redes de sensores", *VI Jornadas de Ingeniería Telemática (JITEL07)*, pp. 401-408, 2007.
NICS Lab. Publications: https://www.nics.uma.es/publications

Análisis de Primitivas Criptográficas para Redes de Sensores

Cristina Alcaraz, Rodrigo Roman, Javier López
Departamento de Lenguajes y Ciencias de la Computación
Universidad de Málaga
E-mail: {alcaraz, roman, jlm}@lcc.uma.es

Abstract *Security in wireless sensor networks is very limited due to highly-constrained hardware of sensor nodes. To protect services is necessary to use secure foundations, known as security primitives, like part of a protocol. Theses primitives must assure at least confidentiality in the communication channel, authentication of the peers involved in an information exchange, and integrity of the messages. There are many primitives such as symmetric encryption, hash functions and public key cryptography, but not all of them can be supported by sensor nodes since require high resource levels, for example memory. This paper contains a deep analysis of available and suitable security primitives for sensor nodes, as well as an analysis of hardware and software implementations. Besides, it has been developed an experiment with two implementations, and it has been created a new and improved version using the optimizations of each.*

1. Introducción

Una red de sensores inalámbrica está compuesta por la agrupación de dispositivos pequeños (nodos sensores) que interaccionan entre sí para alcanzar un objetivo común. Su tarea principal es la de monitorizar un evento físico (aire, humedad, luz, etc.), aunque también pueden enviar o recibir mensajes de avisos, o recibir mensajes de consultas (estado de la red o una determinada propiedad) de alguna estación base, la cual es un dispositivo con mayores recursos que los nodos sensores, y funciona como mediador entre éstos y el usuario final.

Este tipo de red presenta muchos problemas en la parte relacionada con la seguridad, y más aún, cuando esta tecnología, tan reciente, es muy demandada para múltiples aplicaciones, independientemente del tipo de información que se gestione en ella. De hecho, uno de los desafíos propuestos por la comunidad científica [1] es proveer seguridad en todos los niveles, desde la información hasta protocolos y servicios. Sin embargo, proveer y garantizar seguridad no es una tarea sencilla cuando la naturaleza de los nodos está muy limitada en cómputo, almacenamiento y energía. Por ejemplo, el nodo Telosb [2] posee un microcontrolador a 8Mhz, con 48kB de ROM y 10kB de RAM, y el nodo Micaz [3] posee un microcontrolador a 8Mhz, con 128kB de ROM y 4kB de RAM.

Debido a que el principal soporte para la seguridad son las primitivas criptográficas, en este artículo se analizan las que existen actualmente, y aquellas que pueden ser soportadas por los nodos sensores. También, es importante saber qué tipo de implementación (hardware o software) debe aplicarse, ya que dependiendo de la arquitectura del nodo, puede ser soportada o no, y qué tipo de investigaciones se están desarrollando en la actualidad. A parte de este análisis, se ha desarrollado un experimento con dos tipos de implementaciones asimétricas (TinyECC y WMECC), en los nodos Telosb y Micaz, para cuantificar la sobrecarga en tiempo y espacio al ejecutar las primitivas en dichos nodos. Además, se ha obtenido una implementación optimizada (TinyWMECC) por la unificación de aquellas partes optimizadas de cada una de las implementaciones anteriores.

El artículo está organizado de la siguiente manera: en la sección 2 se describen los motivos por los que es necesario garantizar seguridad en este tipo de redes, en la sección 3 se analizan las características de las primitivas criptográficas existentes, identificando las que mejor se ajustan a los nodos sensores, en la sección 4 y 5 se muestra un análisis de implementaciones hardware y software, respectivamente, de las primitivas apropiadas para esta red, en la sección 6 se encuentran las conclusiones, y por último, los agradecimientos.

2. Necesidad de Seguridad en las Redes de Sensores

Proteger lógicamente (ataques pasivos o activos) y físicamente (daños en la arquitectura física) a los nodos, ante malas acciones de adversarios, no es una tarea sencilla. El número de ataques presentes en estos tipos de redes, en comparación con las convencionales, es mayor al existir vulnerabilidades en los nodos, en la

comunicación y en el entorno. Los motivos por los que existen estas debilidades son, en primer lugar, las restricciones hardware y software de los nodos, que dificultan la incorporación de mecanismos seguros. En segundo lugar, el tipo de comunicación (inalámbrica) donde cualquier dispositivo con antena puede acceder, y por último, al ser un medio distribuido en el que ofrecer un servicio implica la cooperación de todos los nodos, cualquier fallo en un nodo podría interrumpir la continuidad del servicio.

Para garantizar seguridad en cualquier instante de tiempo, es necesario utilizar los elementos criptográficos básicos, a fin de proporcionar protección en la información, protocolos y servicios de red. Estos elementos son las primitivas de Criptografía de Clave Simétrica (SKC), de funciones hash, y de Criptografía de Clave Pública (PKC), que se pueden considerar como el "alma" de cualquier protocolo de seguridad. Estas primitivas aseguran confidencialidad e integridad (evitar que individuos sin permiso puedan realizar lecturas y/o alteraciones en los paquetes), y autentificación de ambas partes de la comunicación. Aunque, éstas no garantizan la protección absoluta, es necesario utilizarlas, ya que sin ellas no existiría seguridad.

3. Requisitos de las Primitivas de Seguridad

Cada primitiva posee unas propiedades particulares, y obliga al sistema que la vaya a soportar a cumplir unas exigencias (capacidad de cómputo o memoria). Por consiguiente, no todas pueden ser soportadas por los nodos sensores, y por lo tanto, el objetivo de esta sección es analizar e identificar las primitivas más apropiadas para las redes de sensores.

3.1. Primitivas de Criptografía de Clave Simétrica

En SKC, las primitivas hacen uso de una misma clave tanto para encriptar como para desencriptar. En esta sección, se analizarán algoritmos sencillos y apropiados para dispositivos con recursos restringidos, clasificados en algoritmos de cifrado de bloque y de flujo.

En el cifrado de flujo, la entrada de datos a cifrar posee una longitud variable. Para poder cifrar un mensaje es necesario una transformación de bits mediante la utilización de una clave y un vector de inicialización. Estos dos elementos generan un flujo de clave pseudoaleatorio que será combinado (XOR) bit a bit con el flujo de entrada de datos. Un ejemplo de algoritmo de cifrado de flujo que destaca por su sencillez es el RC4 [4], el cual utiliza operaciones como sumas e intercambios, y otras a nivel de bits como AND y XOR. Dichas operaciones se realizan sobre bloques de 8 bits, con lo cual, no requiere mucha memoria para operar. Es importante usar adecuadamente RC4 para evitar problemas como los existentes en el protocolo WEP [5], al no realizar correctamente la inicialización.

En el cifrado de bloque, la entrada de datos posee un tamaño de longitud fija, la cual va a sufrir una serie de transformaciones para obtener un mensaje cifrado de igual longitud que la entrada. Estas transformaciones se consiguen mediante el uso de una clave y un modo de operación (Electronic Codebook (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB), Output Feedback (OFB), o Counter (CTR)). No existe una clara distinción entre cifrado de flujo y cifrado de bloque, ya que un cifrado de bloque puede actuar en modo "cifrado de flujo" que opera a nivel de bits en lugar de a nivel de bloques.

Skipjack [6] es un algoritmo de cifrado de bloques de 64 bits, rápido y simple. Encripta bloques de datos de 4 palabras utilizando una clave de 80 bits, y alcanzando 32 iteraciones. Para las transformaciones emplea dos reglas (A y B), las cuales se pueden ver como un registro de desplazamiento con retroalimentación lineal y permutaciones G (cifrado Feistel de cuatro iteraciones), que se van aplicando alternadamente. El planificador de clave es sencillo, ya que repite la clave tantas veces como sea necesario para completar el buffer. Sin embargo, la longitud de la clave empleada es muy pequeña.

Un algoritmo algo más complejo que el anterior es el RC5 [7], el cual puede poseer un tamaño de bloque, un tamaño de clave y un número de iteraciones variable. No obstante, se recomienda utilizar un bloque de 64 bits, una clave de 128 bits y 20 iteraciones. El algoritmo está compuesto por tres operaciones (suma de enteros, XOR y rotación variable) sobre dos registros de $b/2$ bits. Un algoritmo muy similar a RC5 es el RC6 [8], que se distingue por la realización de multiplicaciones enteras y la utilización de cuatro registros de tamaño $b/4$ bits. El tamaño de clave puede ser 128, 192 y 256 bits, el tamaño de bloque de 128 bits y 20 iteraciones.

El algoritmo Estándar de Encriptación Avanzada (AES) o Rijndael [9], tiene un bloque de 128 bits, y el número de interacciones (10, 12, 14) dependerá del tamaño de clave que puede ser de 128, 192 o 256 bits. Todas las operaciones en AES se desarrollan sobre un matriz de bytes de 4×4 y sobre un campo finito. Para encriptar un mensaje es necesario pasar por cuatro estados en cada iteración (excepto la última): AddRoundKey (la clave es combinada mediante un XOR con el estado), SubBytes (cada byte en el estado es reemplaza-

do con su entrada tomada de una tabla predeterminada de 8 bits), ShiftRows (los bytes de cada fila del estado son desplazados cíclicamente hacia la izquierda) y MixColumns (cada columna del estado es multiplicada con un polinomio determinado $c(x)$).

Por último, el algoritmo de cifrado Twofish [10] es muy similar al AES, y usa un bloque de 128 bits, con una clave de 256 bits y 16 iteraciones. Twofish utiliza cuatro operaciones biyectivas diferentes, posee un planificador de claves complejo, se basa de las denominadas S-boxes de 8×8 bits de claves dependientes, y emplea algunos de los elementos de otras familias de cifrado, como por ejemplo, la transformación pseudo-Hadamard de la familia SAFER, o la matriz de de distancia máxima separable (MDS) de 4×4 sobre $GF(2^8)$.

3.2. Primitivas de Funciones Hash

Las funciones hash se utilizan para comprimir datos de entrada con longitud variable (e-bits) a uno con un tamaño fijo (s-bits), conocido con el nombre de valor hash h. Estas funciones deben de satisfacer dos propiedades principales: (1) debe ser extremadamente complicado encontrar un mensaje m, tal que $hash(m) = h$, y (2) debe ser difícil encontrar dos mensajes m_1 y m_2, tal que $hash(m_1) = hash(m_2)$. Un ejemplo, es la función hash SHA-1 [11], la cual tiene $e = 512$ bits y $s = 160$ bits, utilizando operaciones XOR, AND, OR, NOT, sumas, y rotaciones. Como la probabilidad de colisión tiende a tener una complejidad del orden 2^{63}, se recomienda la función hash SHA-256 con $e = 512$ bits y $s = 256$ bits. También, existen otras funciones como RIPEMD-160 [12], con $e = 512$ bits y $s = 160$ bits, que usa operaciones de rotación, permutación y desplazamiento.

Otras primitivas criptográficas, como por ejemplo, los Códigos de Autentificación de Mensajes (MAC), utilizan las funciones hash para conseguir integridad en los mensajes y autenticidad. Aunque también pueden ser obtenidas mediante el modo de operación CBC-MAC perteneciente a SKC.

3.3. Primitivas Criptográficas de Clave Pública

La PKC, o criptografía de clave asimétrica, es apropiada en canales de comunicación donde se requiere autentificación e integridad de los datos. Se basa de la utilización de dos claves (privada y pública). La clave privada es sólo conocida por su propietario, y la clave pública es conocida por toda la red. Ambas están relacionadas matemáticamente, pero inferir una a partir de la otra supone un cálculo impracticable. Desafortunadamente, el coste de computación requerido en sus primitivas, dificulta su aplicación en dispositivos con altas restricciones, como los nodos sensores.

A pesar de que el algoritmo asimétrico más popular es el RSA [13] al ofrecer operaciones de encriptación y firma digital, su cálculo operacional es bastante alto, y aplicarlo en contextos restringidos como en las redes de sensores supone un mayor coste. Por estas razones, se recomiendan otros algoritmos más sencillos y adecuados para este tipo de redes, como es la Criptografía de Clave Elíptica (ECC) [14]. Ésta se basa de conceptos algebraicos relacionados con curvas elípticas cuya estructura se corresponde con la ecuación $y^2 = x^3 + ax + b$ sobre un campo finito $\mathbb{F}_p$ o $\mathbb{F}_{2^m}$. Dado a y c, la seguridad de este tipo de criptografía radica en la computación de la ecuación $a^b = c$, conocido como el problema del logaritmo discreto. Su operación básica es la multiplicación en punto escalar, la cual puede ser obtenida realizando sumas y desplazamientos repetidamente, y además, se requiere de la inversión de un entero calculado sobre coordenadas afines. Debido a que el coste computacional es bastante elevado, se han desarrollado algunas optimizaciones, usando el método de Shamir para reducir el tiempo de verificación, o coordenadas proyectivas para evitar las operaciones de inversión.

El tamaño de las claves en ECC es significativamente menor que en RSA, proveyendo la misma seguridad con menos recursos. Además, ECC posee dos primitivas, una para establecer una clave compartida mediante el protocolo Curva Elíptica de Diffie-Hellman (ECDH), y otra para la generación (una multiplicación de punto) y verificación (dos multiplicaciones de punto) de firmas digitales mediante la Curva Elíptica DSA (ECDSA).

Otro algoritmo asimétrico es Rabin [15], un esquema rápido al encriptar datos utilizando la potencia al cuadrado. Su seguridad se encuentra en la dificultad de resolver la raíz cuadrada del cifrado, similar al problema de la factorización de primos grandes, como en el RSA. Aunque, la desencriptación genera cuatro salidas, tres de ellas falsas y una correcta, lo que supone una complejidad mayor.

Por otro lado, tanto el algoritmo NtruEncrypt [16] como su correspondiente de firma digital, conocido como NtruSign, están basados en conceptos aritméticos. Concretamente, en un anillo polinomial $R = \mathbb{Z}(x)/((x^N - 1), q)$, cuyas operaciones básicas son multiplicaciones polinomiales, que lo hacen más rápido que otros esquemas. Su seguridad se encuentra en resolver el problema del vector más corto y el más cercano.

Por último, los criptosistemas de clave pública multivariada, también conocido como $\mathcal{MQ}$-esquemas [17] son uno de los más recientes, y se caracterizan por su velocidad de generación de firmas digitales. Su

seguridad se encuentra en resolver $w = V^{-1}(z) = (\omega_1, ..., \omega_n) \in K^n$, dado un mapa polinomial cuadrático $V = (\gamma_1, ..., \gamma_m) : K^n \rightarrow K^m$. No obstante, existe un coste de almacenamiento en memoria considerable, es decir, se necesita reservar 879 bytes para la clave privada, y 8680 bytes para la clave pública.

3.4. Primitivas de Seguridad Apropiadas

Como se ha comentado en las secciones anteriores, existe una amplia variedad de primitivas de seguridad, aunque no todas pueden ser compatibles con las limitaciones de los nodos. Una de las más adecuadas dentro de las primitivas de SKC es RC4, por su tamaño de bloque y sencillez de sus operaciones. RC4 es apropiado para cualquier microcontrolador con altas restricciones. También el Skipjack se adecúa bastante a microcontroladores limitados pero con una arquitectura de 16 bits, al tener un diseño simple tanto en sus operaciones como en la planificación de claves.

Sin embargo, RC5 y RC6 no son tan adecuados como los anteriores, ya que sus registros son de 32 bits y requieren demasiadas iteraciones (20), aunque sus bloques de construcción sean simples. También, son menos adecuados AES y Twofish por su complejidad, a pesar de tener pocas iteraciones y poder calcular algunas de las operaciones en registros de 8 bits (rápidas de operar).

Con respecto a las primitivas hash, todos los algoritmos nombrados han sido optimizados para procesadores de 32 bits, con lo cual no son apropiados para microcontroladores de 8 o 16 bits. Por otro lado, los bloques de construcción son simples, y por lo tanto, el software resultante es pequeño y rápido. De todas formas, sería interesante desarrollar funciones hash en microcontroladores de 8 y 16 bits.

En PKC, todas las primitivas presentan una complejidad considerable, que las hace inadecuadas para microcontroladores restringidos. Sin embargo, éstas poseen ciertas ventajas, como en ECC, el cual alcanza la misma seguridad que cualquier otro criptosistema de PKC, con una clave mucho menor. El algoritmo NTRUEncrypt ejecuta operaciones de encriptación o desencriptación, y el esquema Rabin, encriptación y verificación, muy veloces. El $\mathcal{MQ}$-esquema genera firmas digitales en tiempos óptimos, y además, si se consigue almacenar las claves en RAM, se asegura un excelente tiempo de verificación.

4. Implementaciones Hardware de Primitivas de Seguridad

Una forma óptima de aplicar primitivas de seguridad se conseguiría mediante el uso de dispositivos hardware, ya que se obtendría mayor eficiencia, rapidez, y además, se podrían soportar algoritmos criptográficos más complejos, en lugar de usar implementaciones software.

4.1. Soporte Hardware Existente

Actualmente, existen varias implementaciones, una de ellas, se encuentra en el estándar 802.15.4 (implementado por el transceptor CC2420). Este estándar incluye primitivas criptográficas, como es el AES, el cual está compuesto por AES-CTR, por AES-CBC-MAC que combina AES con el modo de operación CBC-MAC, y por AES-CCM con un tamaño MAC de 64 bits para proveer confidencialidad y autentificación. Aún así, existen varios problemas a tener en cuenta [18], por ejemplo, el AES-CTR no es capaz de detectar ataques de reenvío de paquetes, pudiendo derivar en otros más serios como el de Denegación de Servicio (DoS). Además, los paquetes ACK pueden ser fácilmente falsificados al no estar protegidos por un MAC.

Para gestionar las entradas y las salidas, los transceptores poseen una lista de control de acceso (ACL) que contiene el tipo de seguridad establecido en una comunicación. De esta forma, cuando el receptor reciba un paquete, podrá aplicar la seguridad que se encuentra especificada en la lista.

Sin embargo, estas implementaciones hardware suelen tener problemas. Por ejemplo, el CC2420 sólo posee dos entradas en su ACL, por lo que sólo provee funcionalidad a dos de sus vecinos.

Por otro lado, es posible conseguir un nivel de paralelismo en el procesamiento de primitivas, mediante instrucciones SIMD (extensión MMX) pertenecientes a la familia del microcontrolador PXA27X de Intel. Esta extensión está compuesta de 16 registros de datos de 64 bits, y de 8 registros de control de 32 bits, permitiendo el cómputo de múltiples operaciones en una misma instrucción. Concretamente, AES posee un paralelismo interno muy elevado, así que es un candidato perfecto para optimización. En cambio, la transformación pseudo-Hadamard de Twofish dificulta el paralelismo.

4.2. Soporte Hardware en Investigación

La comunidad científica está intentando balancear la carga de ejecución de primitivas criptográficas y demás

tareas, en un nodo con altas limitaciones. Actualmente, existen varios prototipos, pero nada tangible. Lo ideal sería disponer de un dispositivo que sea externo o esté adherido al microcontrolador, pero que conjuntamente, reduzca el cómputo de cualquier operación.

Desde que se demostró la posibilidad de aplicar PKC (ver sección 5.2) en redes de sensores mediante ECC, ha existido un gran interés en diseñar dispositivos que la soportasen. De hecho, Wolkerstorfer et. al. [19] y Kumar junto con Paar [20] desarrollaron un chip integrado para generar firmas digitales usando ECDSA. El primer chip opera a 68.5Mhz, tiene 23000 puertas implementadas bajo 0,35μm en tecnología CMOS, siendo capaz de computar una multiplicación de punto sobre el campo finito $\mathbb{F}_{2^{191}}$ en 6,67ms. En cambio, el chip de Kumar y Paar opera a 13Mhz, posee alrededor de 12000 puertas, y computa una multiplicación de punto sobre el campo $\mathbb{F}_{2^{131}}$ en 18ms. A pesar de que existe alto coste computacional en ambas implementaciones, los nodos con microcontroladores PIC18F6720, PXA271, y ARM920T son capaces de soportarlos.

Por otro lado, se encuentra la optimización obtenida por Gaubatz et. al. [21], los cuales tuvieron en cuenta las reducciones modulares, los costes involucrados en las inversiones, e implementaron todas las primitivas aritméticas en un chip especial denominado "bitserial fashion". El dispositivo desarrollado opera a 500kHz, con 18720 puertas bajo 0,13μm en tecnología CMOS, y opera sobre el campo $\mathbb{F}_{p_{100}}$ para la multiplicación de punto escalar. Requiere de 410ms para generar una firma digital con ECDSA o desencriptar mensajes con Curva Elíptica de Menezes Vanstone (ECMV), mientras que para verificar o encriptar con ECMV requiere de 820ms, consumiendo en general menos de 400μW. Una mejora surge en el año 2006, donde Batina et. al. [22] presentan un procesador de curva elíptica con una unidad lógica aritmética modular (MALU) para realizar sumas modulares y multiplicaciones, obtener la potencia al cuadrado a partir de mutliplicaciones, y anular las inversiones usando coordenadas proyectivas. El chip opera a 500kHz, con 8104 puertas bajo 0,13μm en tecnología CMOS, computando una operación de multiplicación de punto sobre $\mathbb{F}_{2^{131}}$ en 115ms, y consumiendo menos de 30μW.

También, Gaubatz et. al. presentaron en [21] una implementación para soportar la primitiva asimétrica Rabin, con menos de 17000 puertas, encriptando y desencriptando mensajes en 2.88ms, y consumiendo una media de 148,18μW. Sin embargo, requería de 1.089s para las operaciones de desencriptación y generación de firma digital. Los mismos autores en [21] propusieron una implementación que soportara la primitiva asimétrica NtruEncrypt, con 3000 puertas, operando a 500kHz, y consumiendo alrededor de 20μW. Para verificar y encriptar mensajes requería de 58ms, para desencriptar 117ms, y para generar una firma digital 234ms.

Por último, Yang et. al. [23] presentaron una implementación para criptosistemas de clave pública multivariada, operando a 100kHz con 17000 puertas bajo 0,25μm en tecnología CMOS. Está pensado para etiquetas de RFID, y sólo puede generar firmas digitales (en 44ms) mediante el método Lanczos, y consumiendo alrededor de 25μA.

5. Implementaciones Software de Primitivas de Seguridad

Pese a que los microcontroladores de los nodos sensores están muy limitados en recursos, en esta sección se va a mostrar que éstos pueden ser capaces de computar primitivas criptográficas a nivel de software.

5.1. Criptografía de Clave Simétrica y Funciones Hash

El objetivo principal en la computación de primitivas SKC y funciones hash es la de alcanzar un tiempo de encriptación, tal que éste no deba ser superior al tiempo de transmisión de un byte por radio. Por ejemplo, si el nodo posee un transceptor CC1000 con 19.2kbps, el tiempo de transmisión es alrededor de 420μs, o para un transceptor CC2420 con 250kbps, el tiempo de transmisión es aproximadamente de 32μs.

Las primeras investigaciones realizadas para analizar la sobrecarga de encriptación de primitivas de clave simétrica y hash en microcontroladores fue en el año 2003 por Ganesan et. al. [24]. Ellos demostraron que encriptar un texto plano de 1 byte con RC5 requería un tiempo de 26μs, con IDEA 21μs y con RC4 6μs. También, analizaron que a pesar de que la función hash SHA-1 necesitaba 7777μs para comprimir 64 bytes, el espacio reservado de memoria no superaba los 4000 bytes.

Hasta el siguiente año no se llegó a confirmar la posibilidad de utilizar las primitivas criptográficas en redes de sensores, con la introducción del paquete TinySec [25] implementado en nesC (lenguaje orientado a componentes), y funcionado sobre el Sistema Operativo TinyOS 1.x, y sólo en los nodos Mica y Mica2. Este paquete provee primitivas como: Skipjack (tiempo de encriptación de 48μs), y un optimizado RC5 (tiempo de encriptación de 33μs). Para conseguir integridad en los datos, éste no utiliza funciones hash, sino CBC-MAC.

En 2006, Wei Law et. al. [26], y Jun Chio y Song [27] volvieron a analizar la sobrecarga de encriptación

observando, que por ejemplo, tanto Twofish como AES consiguen un tiempo de encriptación medio de 50μs, y Skipjack (optimizado) necesitaba 25μs. Con respecto al tamaño medio de memoria, ellos analizaron que RC4 era el algoritmo más optimizado al reservar tan sólo 428 bytes. En cambio, Skipjack necesitaba de 2600 bytes de ROM, y el resto de algoritmos alrededor de 8000 bytes.

5.2. Criptografía de Clave Pública

Hasta en el año 2004, la posibilidad de usar primitivas de clave asimétrica en las redes de sensores se consideraba imposible. La causa que hizo cambiar de idea, fueron los estudios realizados por Gura et. al. [28], los cuales determinaron que mediante ECC era posible implementar PKC. De hecho, Malan et.al. [29] presentaron la librería EccM 2.0 con las primeras primitivas de ECC sobre el campo $\mathbb{F}_{2^p}$, con 163 bits de tamaño de clave, y un tiempo medio de 34s para computar sus operaciones. Sin embargo, 34s era un periodo de tiempo de ejecución considerable, por eso fue optimizado por Gura et. al. [28] al reducir las inversiones con coordenadas proyectivas y utilizando primitivas ECC sobre el campo $\mathbb{F}_p$ para mejorar el rendimiento de su módulo.

Estas optimizaciones y otras fueron implementadas por Liu y Ning con TinyECC [30] (actualmente se encuentra la última versión-0.3 soportada por Micaz, Telosb e Imote2), y Wang y Li con WMECC [31] sobre el Sistema Operativo TinyOS. El diseño de sus implementaciones presenta ciertas características en común, como por ejemplo: ambos hacen uso del $p = 2^n - c$, como primos pseudo-Mersenne, con el fin de obtener una reducción en las multiplicaciones y cuadrados modulares. Además, ambos aplican coordenadas proyectivas, hacen uso del método de ventana deslizante para realizar agrupaciones de escalares y para precomputar las multiplicaciones de punto escalar, y usan el método de Shamir para reducir el tiempo de verificación mediante la multiplicación simultánea de puntos. No obstante, ambos presentan ciertas discrepancias en sus diseños, que les hace ser un tanto diferentes. Concretamente, WMECC usa una mezcla de representaciones Jacobianas (X, Y, Z, aZ^4) y coordenadas afines (X, Y), aplica un valor de ventana pequeño, tanto para el método de ventana deslizante $s = 4$, como para el método de Shamir $w = 1$, con el fin de evitar la precomputación. En cambio, TinyECC utiliza representaciones Jacobianas (X, Y, Z), ejecuta la fase de precomputación para inicializar el sistema ECDSA, y utiliza un método denominado múltiplicación híbrida para gestionar múltiples tamaños de claves.

El cuadro 1 y 2 muestran los resultados obtenidos por un experimento realizado con 20 iteraciones sobre los nodos Micaz y Telosb. Tales resultados representan la sobrecarga (espacio y tiempo) generada por la implementación secp160r1 correspondiente a TinyECC y a WMECC. Dicha implementación está basada en el dominio de curvas elípticas siguiendo las recomendaciones del Grupo de Criptografía Eficiente para Estándares (SECG) [32].

	TinyECC		WMECC	
	Micaz	Telosb	Micaz	Telosb
ROM	28266	26048	57982	46156
RAM	2306	2327	1685	1657
Inic. ECC	1.837s	-	1.809s	1.744s
Inic. ECDSA	3.550s	5.225s	0s	0s
Gen. Clave Pub.	1.788s	-	1.261s	1.425s
Firma Digital	1.916s	4.361s	1.348s	1.498s
Verificación	2.431s	5.448s	2.017s	2.207

Cuadro 1: Sobrecarga en TinyECC y WMECC

	TinyWMECC	
	Micaz	Telosb
ROM	29734	25774
RAM	1643	1599
Inic. ECC	1.809s	1.744s
Inic. ECDSA	0s	0s
Gen. Clave Pub.	1.261s	1.425s
Firma Digital	1.348s	1.498s
Verificación	2.019s	2.209s

Cuadro 2: Sobrecarga en TinyWMECC

Las dos primeras filas en ambos cuadros representan el espacio ocupado en la ROM y RAM del sistema por las primitivas y el test de programa, y el resto de filas representan el tiempo invertido en la ejecución de éstas. Observando, el cuadro 1 se puede analizar, por un lado, que el Telosb es algo más ineficiente que el Micaz, y por otro lado, que el paquete WMECC es mejor en rendimiento que el TinyECC, ya que no existe un proceso de inicialización de ECDSA, y el tiempo de generación de firma digital y su correspondiente verificación es rápida. Sin embargo, como WMECC utiliza una función SHA-1 no optimizada y compleja, reduce el espacio libre de memoria de instrucciones del Telosb, impidiendo la ejecución de otras aplicaciones. En cambio, TinyECC se caracteriza por una función SHA-1 optimizada. Por estas razones, el cuadro 2 muestra la sobrecarga (espacio y tiempo) generada por la implementación TinyWMECC, la cual se puede considerar como un híbrido de las optimizaciones de ambas implementaciones. Esta nueva versión ha sido posible por las características inherentes del lenguaje orientado a componentes, que ha permitido extraer una componente de

TinyECC para añadirla en el código de WMECC.

Además, gracias a la sugerencia realizada por los autores de este artículo, el paquete WMECC ha sido actualizado para contemplar la nueva versión, TinyWMECC.

6. Conclusiones

A pesar de las altas limitaciones presentes en los nodos sensores, y las vulnerabilidades que poseen estos tipos de redes, existe una forma de proteger la comunicación y el medio, mediante primitivas criptográficas. Sin embargo, el uso de una primitiva en un nodo sensor está determinado por varios parámetros, como por ejemplo, la sobrecarga espacial y temporal que puede producir ésta. Por ello, se ha realizado un análisis de las características de cada una de las primitivas, identificando cuáles son las más adecuadas para un nodo sensor.

Por otro lado, usar PKC en redes de sensores se puede considerar un hecho y no un mero concepto teórico, debido a que la ECC aporta mejor rendimiento en cómputo y almacenamiento de claves que otros criptosistemas como el RSA. Además, trabajar en el dominio de curvas elípticas, y concretamente, usando ECDSA proporciona a la red operaciones relacionadas con firmas digitales, y esto a su vez, permite la posibilidad de dotar al sistema de servicios correspondientes a Infraestructuras de Clave Pública [33]. Sin embargo, no todas las primitivas PKC se reducen a ECC, sino que dependiendo del tipo de arquitectura del nodo y del rol que desempeña en la red, pueden usarse otros tipos de criptosistemas. Un ejemplo, sería el $\mathcal{MQ}$-esquema, en el caso de que el nodo tuviera suficiente recursos de memoria, como una estación base.

Por último, se ha realizado un experimento sobre los nodos Micaz y Telosb, para probar dos implementaciones basadas en el dominio de curvas elípticas (TinyECC y WMECC), y bajo el Sistema Operativo orientado a eventos, TinyOS. Se ha observado en función de los resultados, que WMECC es mejor en rendimiento que TinyECC, pero su función hash es bastante compleja. Por estas razones, se han fusionado las optimizaciones de ambas implementaciones, para obtener una nueva versión optimizada.

7. Agradecimientos

Este trabajo forma parte de las investigaciones realizadas en el proyecto Europeo SMEPP (FP6 IST-5-033563) y el proyecto Español CRISIS (TIN2006-09242).

Referencias

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci (2002). *Wireless sensor networks: a survey.* Computer Networks: The International Journal of Computer and Telecommunications Networking, vol. 38, no. 4, pp. 393-422, March 2002.

[2] Moteiv Corporation. http://www.moteiv.com

[3] Crossbow Technology, Inc. Wireless Measurement Systems. http://www.xbow.com

[4] B. Schneier. *Applied Cryptography, 2nd edition.* Wiley, ISBN 0-471-12845-7, 1996.

[5] S. Fluhrer, I. Mantin, A. Shamir. *Weaknesses in the Key Scheduling Algorithm of RC4.* In proceedings of the 8th Annual Workshop on Selected Areas in Cryptography (SAC 2001), Toronto (Canada), August 2001.

[6] NIST-CSRC. *SKIPJACK and KEA Algorithm Specifications, version 2.* 29 May 1998, http://csrc.nist.gov/CryptoToolkit/

[7] R. L. Rivest. *The RC5 Encryption Algorithm.* Proceedings of the 2nd International Workshop on Fast Software Encryption (FSE 1994), Leuven (Belgium), December 1994.

[8] R. L. Rivest, M. J. B. Robshaw, R. Sidney, Y. L. Yin. *The RC6 Block Cipher, v1.1.* August 1998, http://theory.lcs.mit.edu/ rivest/

[9] J. Daemen, V. Rijmen. *The Design of Rijndael.* Springer, ISBN 3-540-42580-2, 2002.

[10] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, N. Ferguson. *The Twofish Encryption Algorithm: A 128-Bit Block Cipher.* Wiley, ISBN 0-471-35381-7, 1999.

[11] D. Eastlake, P. Jones. *US Secure Hash Algorithm 1 (SHA1).* RFC 3174.

[12] H. Dobbertin, A. Bosselaers, B. Preneel. *RIPEMD-160, a strengthened version of RIPEMD.* Proceedings of the 3rd International Workshop on Fast Software Encryption (FSE 1996), Cambridge (UK), February 1996.

[13] R. Rivest, A. Shamir, L. Adleman. *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems.* Communications of the ACM, vol. 21, no. 2, pp. 120−126, 1978.

[14] I. Blake, G. Seroussi, N. P. Smart. *Elliptic Curves in Cryptography*. Cambridge University Press, ISBN 0-521-65374-6, 2000.

[15] M. O. Rabin. *Digitalized Signatures and Public Key Functions as Intractable as Factorization*. Technical Report MIT/LCS/TR-212, Massachusetts Institute of Technology (1979).

[16] J. Hoffstein, J. Pipher, J. H. Silverman. *NTRU: a Ring based Public Key Cryptosystem*. In proceedings of the 3rd Algorithmic Number Theory Symposium (ANTS 1998), Portland (USA), June 1998.

[17] C. Wolf, B. Preneel. *Taxonomy of Public Key Schemes Based on the Problem of Multivariate Quadratic Equations*. Cryptology ePrint Archive, Report 2005/077.

[18] N. Sastry, D. Wagner. *Security considerations for IEEE 802.15.4 networks*. In Proceedings of 2004 ACM Workshop on Wireless security (Wise 2004), Philadelphia (USA), October 2004.

[19] J. Wolkerstorfer. *Scaling ECC Hardware to a Minimum*. In ECRYPT workshop - Cryptographic Advances in Secure Hardware - CRASH 2005. Leuven (Belgium), September 2005. Invited Talk.

[20] S. Kumar, C. Paar. *Are standards compliant elliptic curve cryptosystems feasible on RFID?*. In Proceedings of Workshop on RFID Security, Graz (Austria), July 2006.

[21] G. Gaubatz, J.-P. Kaps, E. Öztürk, B. Sunar. *State of the Art in Ultra-Low Power Public Key Cryptography for Wireless Sensor Networks*. In Proceedings of the 2nd IEEE International Workshop on Pervasive Computing and Communication Security (PerSec 2005), Hawaii (USA), March 2005.

[22] L. Batina, N. Mentens, K. Sakiyama, B. Preneel, I. Verbauwhede. *Low-Cost Elliptic Curve Cryptography for Wireless Sensor Networks*. In Proceedings of the 3rd European Workshop on Security and Privacy in Ad hoc and Sensor Networks (ESAS 2006), Hamburg (Germany), September 2006.

[23] B.-Y. Yang, C.-M. Cheng, B.-R. Chen, J.-M. Chen. *Implementing Minimized Multivariate Public-Key Cryptosystems on Low-Resource Embedded Systems*. In Proceedings of the 3rd International Conference on Security in Pervasive Computing (SPC 2006), York (UK), April 2006.

[24] P. Ganesan, R. Venugopalan, P. Peddabachagari, A. Dean, F. Mueller, M. Sichitiu. *Analyzing and Modeling Encryption Overhead for Sensor Network Nodes*. In Proceedings of the 2nd ACM International Conference on Wireless Sensor Networks and Applications (WSNA 2003), San Diego (USA), September 2003.

[25] C. Karlof, N. Sastry, D. Wagner. *TinySec: a link layer security architecture for wireless sensor networks*. Proceedings of 2nd International Conference on Embedded Networked Sensor Systems (SenSys 2004), Baltimore (USA), November 2004.

[26] Y. W. Law, J. Doumen, P. Hartel. *Survey and Benchmark of Block Ciphers for Wireless Sensor Networks*. ACM Transactions on Sensor Networks, vol. 2, no. 1, pp 65-93, February 2006.

[27] K. Jun Choi, J.-I. Song. *Investigation of Feasible Cryptographic Algorithms for Wireless Sensor Network*. Proceedings of the 8th International Conference on Advanced Communication Technology (ICACT 2006). Phoenix Park (Korea), February 2006.

[28] N. Gura, A. Patel, A. Wander. *Comparing elliptic curve cryptography and RSA on 8-bit CPUs*. In Proceedings of the 2004 Workshop on Cryptographic Hardware and Embedded Systems (CHES 2004), Cambridge (USA), August 2004.

[29] D. J. Malan, M. Welsh, M. D. Smith. *A Public-key Infrastructure for Key Distribution in TinyOS Based on Elliptic Curve Cryptography*. In Proceedings of 1st IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON 2004), Santa Clara (USA), October 2004.

[30] An Liu, Peng Ning, *TinyECC: Elliptic Curve Cryptography for Sensor Networks (Version 0.3)*. http://discovery.csc.ncsu.edu/software/TinyECC/, September 2006.

[31] H. Wang, Q. Li. *Efficient Implementation of Public Key Cryptosystems on MICAz and TelosB Motes*. Technical Report WM-CS-2006-07, College of William & Mary, October 2006.

[32] SECG - Standards for Efficient Cryptography Group. http://www.secg.org/

[33] J. Lopez. *Unleashing Public-Key Cryptography in Wireless Sensor Networks*. Journal of Computer Security, vol 14, no. 5, pp 469-482, 2006.